ENHANCING GENERIC CODE CLONE DETECTION MODEL FOR C BASED APPLICATION

AINUN SYAHIRAH BINTI ADNAN

Bachelor of Computer Science (Software Engineering)

UNIVERSITI MALAYSIA PAHANG

SUPERVISOR'S DECLARATION

I hereby declare that I have checked this thesis and in my opinion, this thesis is adequate in terms of scope and quality for award of degree of Bachelor of Computer Science (Software Engineering).

_____

(Supervisor's Signature)

Full Name : DR. AL- FAHIM BIN MUBARAK ALI

Date :

STUDENT'S DECLARATION

I hereby declare that the work in this thesis is based on my original work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at Universiti Malaysia Pahang or any other institutions.

_____

(Student's Signature)

Full Name : AINUN SYAHIRAH BINTI ADNAN

ID Number : CB16011

Date :

ENHANCING GENERIC CODE CLONE DETECTION MODEL FOR C BASED APPLICATION

AINUN SYAHIRAH BINTI ADNAN

Thesis submitted in fulfilment of the requirements for the award of the degree of Bachelor of Computer Science (Software Engineering)

Faculty of Computer Systems & Software Engineering

UNIVERSITI MALAYSIA PAHANG

MAY 2019

ACKNOWLEDGEMENTS

First of all, I want to thank God for the thanksgiving for giving me the health of my body as long as I prepare this thesis. I also would like to thank all those who helped and support in making this thesis especially for my parents, family, supervisor, lectures and friends.

Therefore, I sincerely want to thank my supervisor, Dr. Al- Fahim Bin Mubarak Ali for giving me a lot of knowledge and guidance throughout I prepare this thesis. Without his helps I am sure that I cannot succeed to complete this thesis. Besides, I would like to thank to my course mates and my roommates that share with their knowledge and perception.

Lastly, I want to give my appreciation to my family because the warm encouragement that has been given to me. I also want to say thank you for those were involved directly or indirectly in this project.

ABSTRAK

Klon kod adalah istilah yang digunakan untuk menggambarkan kod yang digunakan dalam sistem berulang kali. Pada masa ini terdapat empat jenis klon kod, iaitu jenis-1, jenis-2, jenis-3 dan jenis-4, yang dapat dikesan oleh beberapa alat pengesan klon kod. Setakat kualiti sistem berkenaan, klon kod boleh menyebabkan sistem memakan lebih banyak memori untuk menjalankan fungsi, kerana banyak kod yang digunakan berulang kali. Klon kod juga mempengaruhi proses penyelenggaraan sistem. Sekiranya fragmen kod yang disalin mengandungi pepijat, semua kod dengan persamaan dengan fragmen kod yang disalin mestilah diperbaiki satu persatu. Ia mengambil masa yang lama untuk mengekalkan sistem. Aplikasi yang dibangunkan di Java dan C biasanya mempunyai kemungkinan besar klon kod disebabkan penggunaan bahasa-bahasa ini yang melampau dalam pembangunan aplikasi. Oleh itu, objektif utama penyelidikan ini adalah untuk memperbaiki model pengesanan klon kod untuk mengesan klon kod dalam bahasa pengaturcaraan C. Pelbagai model boleh didapati untuk mengesan kod klon yang merupakan model klon generik, model saluran paip generik, model klon bersatu dan model pengesanan klon kod generik. Pengesanan Klon Generik Kod (GCCD) adalah keadaan model seni yang mengesan klon kod sehingga menaip 4 dalam program Java. Proses model ini adalah pra-pemprosesan, pemprosesan, parameterisasi, pengkategorian dan pengesanan padanan. Tujuan penyelidikan ini adalah untuk meningkatkan prototaip untuk mengesan klon kod dalam bahasa pengaturcaraan C. Oleh itu, objektif utama penyelidikan ini adalah untuk meningkatkan prototaip model pengesanan klon generik kod untuk mengesan klon kod dalam bahasa pengaturcaraan C. Kajian ini memberi tumpuan kepada meningkatkan dua proses, iaitu pra-pemprosesan dan transformasi. Untuk menilai penambahbaikan yang dibuat dalam kajian ini, prototaip GCCD dipertingkatkan dan diuji dengan menggunakan set data penanda aras yang dipanggil dataset penanda aras Bellon. Hasil yang diharapkan dari kajian ini ialah prototaip GCCD dapat mengesan kloning bahasa pemrograman C.

ABSTRACT

Code clone is a term used to describe a code used in a system repeatedly. There are currently four types of code clones, namely type-1, type-2, type-3 and type-4, which can be detected by some code clone detection tools. As far as the quality of a system is concerned, the code clone can cause a system to consume more memory to perform a function, due to the many codes that are repeatedly used. The code clone also affects the system maintenance process. If the copied code fragment contains a bug, all code with similarities to the copied code fragment must be fixed one by one. It takes longer to maintain the system. Applications developed in Java and C usually has the largest occurrence of code clone due to the extreme usage of these languages in application development. Therefore, the main objective of this research is to improve the code clone detection model to detect the code clone in the language of C programming. Various models are available to detect a clone code which is a generic clone model, generic pipeline model, unified clone model and a generic code clone detection model. Generic Code Clone Detection (GCCD) is the state of the art model that detects code clone up to type 4 in Java programs. This model's process is pre-processing, processing, parameterization, categorization and match detection. The aim of this research is to improve the prototype for the detection of code clones in the C programming language. Therefore, the main objective of this research is to improve the prototype of the generic code clone detection model to detect the code clone in the language of C programming. This research focuses on improving two processes, namely pre-processing and transformation. In order to evaluate the improvements made in this research, the GCCD prototype is enhanced and tested using a benchmark data set called Bellon's benchmark dataset. The expected result of this research is that the GCCD prototype can detect the C programming language code clone.

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

GCCD	Generic Code Clone Detection
VB	Visual Basic
PDG	Program Dependence Graph
CP	Clone Pair
CC	Clone Class
AST	Abstract Syntax Tree
SWOT	Strength Weakness Opportunity Threat
LOC	Line Of Code

CHAPTER 1

INTRODUCTION

1.1 Introduction

In the process of developing a system, developers typically reuse the code they make into another line of code or other modules in the same system. This practice or behaviour makes it easier for them to speed development process. This method can be called code clone or duplicate code. Code clone is a process of reusing a code repeatedly. Discussion related to code clone has been done by several researcher, most of the current systems, the results showed a fraction that between 20% to 30% of module in system may be cloned (Baker, 1995).

There are 4 types which are **Type-1,** code portion or fragments are identical, except for spacing, layout and comment variations. In clone type 1 also known as exact clone. This is because the different fragments are exact copies of each other. **Type-2**, code portion or fragment are syntactically identical, except for literals, identifiers, types, comments, and layout and whitespace variations. This type is like Type-1 which is have similarity of code portion or fragment to each other, but have more addition for identifiers declared (constants, class, methods, name of variables and so on). **Type-3**, this type is some evolution of type-2, the different of this type is the fragment that are copied from another source code have some added statements, with removal of some statements or some modified statement. **Type-4**, two or more code fragment or portion that have similarity with each other and perform the same computation, but different syntactic variants, called by type-4.

This type is not mandatory that code fragment should be copied from somewhere or different programmer but have same functionality. Otherwise, the category of two code fragment or portion is under Type-4. Most of the current code clone detection tools only detect until Type-3. Meanwhile, the purpose of this study, which is the addition of a function for the prototype of GCCD is to detect the code clone for a C programming language that supports to detect the code clone up to Type-4.

Although removing a code clone has the risk of changing the software structure or framework, however, to find out a code clone in a software, it is an advantage to consider whether it is necessary to make a changes for the code clone. In visual basic, there already have their code clone detection, but the code clone detection tool on this application only detect until Type-3.

Generic code clone detection model (GCCD) is a model used to detect code clone in a system. There are some research that explains the code clone model, the generic code clone model is one of the last models developed by pre-existing research. Previously, a prototype has been developed using this model, GCCD model can detect code clone up to Type-4 compared to other models but it can only detect code clone in Java programming language.

Also, there are several types of models and approaches for detecting code clone. This approach is widely used by other researches in establishing or improving existing code clone detection methods. Below are five (5) approaches of detecting code clone (Al-Fahim, 2015):

i) String based comparison

This approach will detect code clone by comparing source code by text / string that in the line of code in the same fragment.

ii) Metric based comparison

This Approach works by comparing different metrics and gathering into one, and on the basis of similar value, the similarity will detected.

iii) Tree based comparison

In this approach, abstract syntax tree of a system is produced. Then, tree matching technique is applied to detect similar sub trees. When the result come out between two sub trees, the source code of similar sub trees is returned as clone-pair.

iv) Token based comparison

This approach are detecting the code that have token sequence obtained from division the line of source code. The unique of this approach is the characteristic is using hash function.

v) Graph based comparison

The graph based comparison are the approach that detecting code clone by converting the code into the graph version. After several of graph that already detect the similarity, the function of this approach will continue with a clone-pair.

In addition to the approach, the model for code clone detection has a role to unify the tools and also the approaches that will be describe at above. An approach is a way of detecting a code clone by comparing of something. The model is a more complex way of detecting code clone on the system.

There are several model that can be used for code clone detection. The three model that always use for code clone detection, it is generic clone model, generic pipeline clone model, unified clone model, and generic code clone detection model . This research will focusing on enhancing the generic code clone detection model.

REFERENCES

Al-Fahim. (2015). Generic Code Clone Detection Model for Java Applications, (August).

Baker, B. S. (1995). On Finding Duplication and Near-Duplication in Large Software Systems.

Basit, H. A., Rajapakse, D. C., & Jarzabek, S. (2005). Beyond templates: a study of clones in the STL and some general implications. *Proceedings of the 27th International Conference on Software Engineering*, 451–459. https://doi.org/10.1145/1062455.1062537

Baxter, I. D., Yahin, A., Moura, L., Sant'Anna, M., & Bier, L. (1998). Clone detection using abstract syntax trees. *Proceedings. International Conference on Software Maintenance (Cat. No. 98CB36272)*, 368–377. https://doi.org/10.1109/ICSM.1998.738528

Bellon, S., Koschke, R., Antoniol, G., Krinke, J., & Merlo, E. (2007). Comparison and Evaluation of Clone Detection Tools, *33*(9), 577–591.

Biegel, B., & Diehl, S. (2010). Highly configurable and extensible code clone detection. In *Proceedings - Working Conference on Reverse Engineering, WCRE* (pp. 237–241). https://doi.org/10.1109/WCRE.2010.34

Cordy, J. R., & Roy, C. K. (2011). The NiCad clone detector. In *IEEE International Conference on Program Comprehension* (pp. 219–220). https://doi.org/10.1109/ICPC.2011.26

Dang, Y., Zhang, D., Ge, S., Huang, R., Chu, C., & Xie, T. (2017). Transferring code-clone detection and analysis to practice. *Proceedings - 2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering in Practice Track, ICSE-SEIP 2017*, 53–62. https://doi.org/10.1109/ICSE-SEIP.2017.6

Duala-Ekoko, E., & Robillard, M. P. (2007). Tracking code clones in evolving software. In *Proceedings - International Conference on Software Engineering* (pp. 158–167). https://doi.org/10.1109/ICSE.2007.90

Duala-Ekoko, E., & Robillard, M. P. (2008). Clonetracker: Tool Support for Code Clone Management. In *Proceedings of the 13th international conference on Software engineering - ICSE '08* (p. 843). https://doi.org/10.1145/1368088.1368218

Giesecke, S. (2007). Generic modelling of code clones. *Duplication, Redundancy, and Similarity in Software*, (06301), 1–23.

Göde, N., & Koschke, R. (2009). Incremental clone detection. In *Proceedings of the European Conference on Software Maintenance and Reengineering, CSMR* (pp. 219–228). https://doi.org/10.1109/CSMR.2009.20

Harder, J. (2013). The Limits of Clone Model Standardization, 10–11.

Higo, Y., & Ueda, Y. (2007). Simultaneous Modification Support based on Code Clone Analysis, 262–269. https://doi.org/10.1109/ASPEC.2007.44

Jiang, L., Misherghi, G., & Su, Z. (2007). D ECKARD : Scalable and Accurate Tree-based Detection of Code Clones ∗, (0520320).

Kamiya, T., Kusumoto, S., & Inoue, K. (2002a). CCFinder: A Multilinguistic Token-Based Code Clone Detection System for Large Scale Source Code CCFinder: A Multilinguistic Token-Base Code Clone Detection System for Large Scale Source Code. *Ieee Transactions on Software Engineering*, *28*(28), 654–670.

Kamiya, T., Kusumoto, S., & Inoue, K. (2002b). Review of CCFinder : A Multi-linguistic Token-Based Code Clone Detection System for Large Scale Source Code, *28*(7), 3.

Kapser, C. J., Harder, J., & Baxter, I. (2012). A common conceptual model for clone detection results. *2012 6th International Workshop on Software Clones, IWSC 2012 - Proceedings*, 72–73. https://doi.org/10.1109/IWSC.2012.6227870

Kodhai, E., & Kanmani, S. (2014). Method-level code clone detection through LWH (Light Weight Hybrid) approach. *Journal of Software Engineering Research and Development*, *2*(1), 12. https://doi.org/10.1186/s40411-014-0012-8

Komondoor, R., & Horwitz, S. (2001). Using Slicing to Identify Duplication in Source Code, 40–56. https://doi.org/10.1007/3-540-47764-0_3

Kontogiannis, K. A., Demori, R., Merlo, E., Galler, M., & Bernstein, M. (1996). Pattern matching for clone and concept detection. *Automated Software Engineering*, *3*(1–2), 77–108. https://doi.org/10.1007/BF00126960

Krinke, J. (2001). Identifying similar code with program dependence graphs. *Proceedings Eighth Working Conference on Reverse Engineering*, 301–309. https://doi.org/10.1109/WCRE.2001.957835

Lavoie, T., Eilers-Smith, M., & Merlo, E. (2010). Challenging cloning related problems with GPU-based algorithms. *Proceedings of the 4th International Workshop on Software Clones - IWSC '10*, 25–32. https://doi.org/10.1145/1808901.1808905

Liu, C., Chen, C., Han, J., & Yu, P. S. (2006). GPLAG: detection of software plagiarism by program dependence graph analysis. *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 872–881. https://doi.org/10.1145/1150402.1150522

Livieri, S., Higo, Y., Mazushita, M., & Inoue, K. (2007). Very-Large Code Clone Analysis and Visualization of Open Source Programs Using Distributed CCFinder: D-CCFinder. *Proceedings of the 29th International Conference on Software Engineering (ICSE '07)*.

Mayrand, Leblanc, & Merlo. (1996). Experiment on the automatic detection of function clones in a software system using metrics. *Proceedings of International Conference on Software Maintenance ICSM-96*, 244–253. https://doi.org/10.1109/ICSM.1996.565012

Morshed, M., Rahman, M., & Ahmed, S. (2012). A Literature Review of Code Clone Analysis to Improve Software Maintenance Process. *ArXiv Preprint ArXiv:1205.5615*.

Nguyen, T. T., Nguyen, H. A., Pham, N. H., Al-Kofahi, J. M., & Nguyen, T. N. (2009). ClemanX: Incremental clone detection tool for evolving software. *2009 31st International Conference on Software Engineering - Companion Volume, ICSE 2009*, 437–438. https://doi.org/10.1109/ICSE-COMPANION.2009.5071050

Perumal, A., Kanmani, S., & Kodhai, E. (2010). Extracting the similarity in detected software clones using metrics. *2010 International Conference on Computer and Communication Technology, ICCCT-2010*, 575–579. https://doi.org/10.1109/ICCCT.2010.5640465

Rattan, D., Bhatia, R., & Singh, M. (2013). *Software clone detection: A systematic review*. *Information and Software Technology* (Vol. 55). Elsevier B.V. https://doi.org/10.1016/j.infsof.2013.01.008

Roy, C. K., & Cordy, J. R. (2007). A Survey on Software Clone Detection Research. *Queen's School of Computing TR*, *115*, 115. https://doi.org/10.1.1.62.7869

Sasaki, Y., Yamamoto, T., Hayase, Y., & Inoue, K. (2010). Finding file clones in FreeBSD ports collection. *Proceedings - International Conference on Software Engineering*, 102–105. https://doi.org/10.1109/MSR.2010.5463293